UDC: 004.931; 004.932.2 DOI: https://doi.org/10.15407/jai2021.02.42

RECOGNITION AND CALCULATION OF OBJECTS IN IMAGES USING YOLOv3 ARCHITECTURE

V. Hrabovskyi[1], O. Kmet[2]
[1,2] Ivan Franko National University of Lviv, Ukraine
107, Gen. Tarnavskoho str., Lviv, 79017

[1] http://orcid.org/0000-0003-2047-8688

Abstract. Program that searches for five types of fruits in the images of fruit trees, classifies them and counts their quantity is presented. Its creation took into account the requirement to be able to work both in the background and in real time and to identify the desired objects at a sufficiently high speed. The program should also be able to learn from available computers (including laptops) and within a reasonable time.

In carrying out this task, the possibilities of several existing approaches to the recognition and identification of visual objects based on the use of convolutional neural networks were analyzed. Among the considered network architectures were R-CNN, Fast R-CNN, Faster R-CNN, SSD, YOLO and some modifications based on them. Based on the analysis of the peculiarities of their work, the YOLO architecture was used to perform the task, which allows the analysis of visual objects in real time with high speed and reliability.

The software product was implemented by modifying the YOLOv3 architecture implemented in TensorFlow 2.1. Object recognition in this architecture is performed using a trained Darknet-53 network, the parameters of which are freely available. The modification of the network was to replace its original classification layer. The training of the network modified in this way was carried out on the basis of Transfer learning technology using the Agrilfruit Dataset. There was also a study of the peculiarities of the learning process of the network under the use of different types of gradient descent (stochastic and with the value of the batch 4 and 8), as a result of which the optimal version of the trained network weights was selected for further use.

Tests of the modified and trained network have shown that the system based on it with high reliability distinguishes objects of the corresponding classes of different sizes in the image (even with their significant masking) and counts their number. The ability of the program to distinguish and count the number of individual fruits in the analyzed image can be used to visually assess the yield of fruit trees.

Keywords: pattern recognition, object identification, deep neural networks, convolution neural networks, YOLOv3, Darknet-53, Agrilfruit Dataset.

РОЗПІЗНАВАННЯ ТА ПІДРАХУНОК ОБ'ЄКТІВ НА ЗОБРАЖЕННІ ЗА ДОПОМОГОЮ АРХІТЕКТУРИ YOLOv3

В. А. Грабовський[1], О. Я. Кметь[2]
[1, 2] Львівський національний університет імені Івана Франка, Україна
вул. Ген. Тарнавського, 107, м.Львів, 79017

[1] http://orcid.org/0000-0003-2047-8688

Анотація. Представлена програма, яка здійснює пошук п’яти видів плодів на зображеннях фруктових дерев, класифікує їх та підраховує їх кількості. При її створенні була врахована вимога можливості роботи як у фоновому режимі, так і в режимі реального часу та ідентифікувати потрібні об’єкти з достатньо високою швидкістю. Програма також повинна мати можливість навчатися за допомогою доступних комп’ютерів (включаючи ноутбуки) і в межах розумного часу.

При реалізації поставленої задачі у роботі були проаналізовані можливості декількох існуючих підходів до розпізнавання та ідентифікації візуальних об’єктів на основі використання згорткових нейронних мереж. Серед розглянутих мережних архітектур були R-CNN, Fast R-CNN, Faster R-CNN, SSD, YOLO та деякі модифікації на їх основі. На підставі проведеного аналізу особливостей їх роботи для виконання поставленої задачі було взято архітектуру YOLO, яка дозволяє проводити аналіз візуальних об’єктів в реальному режимі часу з високою швидкістю та надійністю.

Реалізація програмного продукту була здійснена шляхом модифікації архітектури YOLOv3, реалізованої в TensorFlow 2.1. Розпізнавання об’єктів в цій архітектурі здійснюється з допомогою навченої мережі Darknet-53, параметри якої знаходяться у вільному доступі. Модифікація мережі полягала у заміні її вихідного класифікуючого шару. Навчання модифікованої таким чином мережі здійснено на основі технології Transfer learning з використанням датасету Agrilfruit Dataset. Було проведене дослідження особливостей процесу навчання мережі

за умови використання різних видів градієнтного спуску (стохастичного та зі значенням батчу 4 і 8), в результаті якого був обраний оптимальний варіант ваг навченої мережі для подальшого її використання.

Тестування роботи модифікованої та навченої мережі показало, що система на її основі з високою надійністю розрізняє об'єкти відповідних класів різного розміру на зображенні (навіть зі значним їх маскуванням) та підраховує їх кількість. Здатність програми розрізняти та підраховувати кількість окремих плодів на аналізованому зображенні може бути використана для візуальної оцінки врожайності плодових дерев.

Ключові слова: розпізнавання образів, ідентифікація об'єктів, глибокі нейронні мережі, згорткові нейронні мережі, YOLOv3, Darknet-53, Agrilfruit Dataset.

Вступ

Виявлення та ідентифікація об'єктів – комп'ютерна технологія, пов'язана з комп'ютерним зором та обробкою зображень, яка займається виявленням об'єктів певного класу на цифрових зображеннях та відео – останнім часом набуває все більш важливого значення завдяки своєму широкому застосуванню в пристроях різноманітного призначення. До сфери виявлення об'єктів віднесене виконання багатьох функціональних операцій, які включають у себе як виявлення наявних на зображенні об'єктів, їх меж та країв, знаходження та ідентифікацію, у т. ч. і живих істот (зокрема – людини, її пози, виявлення обличчя та його ідентифікацію), так і загальний аналіз змісту спостережуваної сцени чи ситуації. Сьогодні без використання засобів такого розпізнавання просто неможливо уявити успішну діяльність у робототехніці, сфері моніторингу безпеки, автономному керуванні різноманітними транспортними засобами, спостереженні за станом довкілля та реагуванні на його зміни (у т. ч. і в надзвичайних ситуаціях), аналізі місця знаходження та результатів роботи безпілотних літальних апаратів, різноманітних систем військового, медичного та іншого призначення.

Більшість сучасних засобів та пристроїв розпізнавання об'єктів і як засіб для виділення особливостей із вхідних зображень (або відео), і як саму мережу виявлення, класифікації та локалізації об'єктів використовують мережі глибокого навчання. Особливо важливого значення при вирішенні проблем розпізнавання набули глибокі нейронні згорткові мережі, які завдяки особливостям своєї будови та функціонування дуже добре "витягують" ознаки із зображення. Саме завдяки цьому вони нині широко застосовуються для вирішення задач класифікації, розпізнавання, сегментації та багатьох інших.

При розробці таких систем значна увага приділяється як створенню нових видів мереж, які забезпечують реалізацію основних задач розпізнавання (до них відносяться насамперед підвищення точності виявлення об'єктів та швидкодії системи), так і зручності їх навчання та кінцевого застосування. Такий підхід зумовив розробку ефективних методів виявлення і розпізнавання об'єктів на основі згорткових нейронних мереж, які загалом можна розділити на методи з використанням як двоступеневого підходу, в якому процес розпізнавання можна охарактеризувати як почергове "грубе" і "тонке" виявлення ознак об'єктів та їх класифікацію різними мережами, так і одноступеневого (де розпізнавання здійснюється однією мережею "в один крок"), які забезпечують ефективну роботу систем виявлення і розпізнавання об'єктів в реальному часі з використанням цифрових фото- та відеоматеріалів [1, 2].

Крім вдосконалення підходів до розпізнавання та створення складних архітектур нейронних мереж, велика увага приділяється також створенню і успішному використанню навчальних тестових наборів, (Caltech [3], KITTI [4], ImageNet [5], PASCAL VOC [6], MS COCO [7], Open Images V5 [8] та інші), використання яких дозволило ефективно навчати глибокі нейронні мережі.

Оскільки навчання складних багатошарових мереж є дуже складним і вартісним процесом і вимагає значних затрат як обчислювальних, так і часових ресурсів, для їх зменшення були розроблені і набули широкого практичного застосування методи використання вже навчених і апробованих мереж на основі підходу transfer learning [9]. Застосування таких методів не вимагає від користувача проведення повного циклу навчання моделі з необхідним при

цьому використанням великих датасетів та потужних обчислювальних систем і дає можливість застосовувати їх на практиці з незначними модифікаціями самої мережі та її навчання на невеликих датасетах і, що головне – з використанням доступної комп'ютерної техніки.

Такий стан справ привів до того, що на сьогодні в галузі комп'ютерного розпізнавання візуальних об'єктів напрацьована низка підходів з використанням різних нейромережевих архітектур, застосування яких у кожному конкретному випадку надає користувачам як певні переваги, так і не вільне від деяких притаманних їм недоліків. Тому при виборі потрібного інструменту для вирішення конкретної часткової задачі користувачеві потрібно враховувати усі притаманні йому нюанси. Не останню роль при цьому відіграє наявність потрібної моделі у вільному доступі, можливість її модифікації під вирішення поставленої задачі без значних затрат, а також навчання створеного на її основі додатку з використанням доступних ресурсів.

Мета роботи та підходи до її вирішення

Метою даної роботи було створити застосунок, який міг би, базуючись на можливостях доступного персонального комп'ютера та наявних у вільному доступі нейромережевих технологій, здійснювати аналіз зображень певного виду з метою знаходження, класифікації та підрахунку окремих об'єктів, що знаходяться на них. Зокрема, створена програма повинна надавати можливість проводити аналіз зображень плодових дерев певних видів (яблук, груш, лимонів, хурми, цукрового яблука) на наявність на них відповідних плодів та підраховувати їх кількість. Результати роботи програми повинні видаватись як у графічному форматі, де кожний знайдений плід має бути відмічений відповідним за його розміром прямокутником з іменною поміткою та ймовірністю його ідентифікації, так і в текстовому, де вказані вид ідентифікованого об'єкта та сумарна кількість усіх знайдених на аналізованому зображенні плодів.

Згідно поставленої задачі, для її реалізації потрібно було використати наявні у вільному доступі інструменти, які для вирішення проблем розпізнавання та ідентифікації застосовують найефективніші на сьогодні структури – глибокі нейронні мережі, зокрема згорткові нейронні мережі.

Для вибору потрібної архітектури моделі були розглянуті деякі популярні нейромережеві підходи, які розроблені для виконання операцій щодо знаходження та класифікації візуальних об'єктів і володіють хорошою швидкодією. Як уже згадувалося, за застосованим підходом до використання глибоких згорткових мереж у пристроях розпізнавання зображень, останні можна розділити на дві категорії – дво- та одноступеневі. До перших (т. з. двоступеневих детекторів) належать пристрої зі структурою R-CNN (Region Convolution Neural Network) [10] та її модифікацією Fast R-CNN [11] й Faster R-CNN [12]. Ідея R-CNN-підходу полягає у вибірковому виявленні набору блоків кандидатів у шукані об'єкти однією мережею, масштабування кожного з них до фіксованого розміру та передачі іншій навченій згортковій мережі для вилучення нею ознак виявлених кандидатів. Саме ж передбачення та розпізнавання здійснюється за допомогою лінійних класифікаторів з використанням виявлених на попередньому етапі ознак. Завдяки розділеному виявленню і класифікації таким методом, досягається досить висока точність розпізнавання, у т. ч. і об'єктів дрібних розмірів, але суттєво зменшується швидкість самого процесу розпізнавання. Саме на збільшення швидкості розпізнавання направлені в основному модифікації і вдосконалення цього підходу – Fast R-CNN й Faster R-CNN.

У одноступеневому підході повністю відмовляються від парадигми розпізнавання двоступеневого методу, суть якої – виявлення пропозиції для розпізнавання та її перевірка проводяться різними мережами; тут застосовують і для виявлення, і для розпізнавання об'єкта одну і ту ж саму згорткову мережу. Такий підхід дозволяє суттєво пришвидшити процес розпізнавання завдяки розпізнаванню кандидатів на виявлення "за один раз однією й тією ж мережею" при вказаному підході досягнута швидкість розпізнавання до 155 кадрів за

секунду. Однак, такий підхід зумовлює деякі труднощі саме в розпізнаванні, особливо малорозмірних об’єктів. І тому саме в напрямку покращення розпізнавання малорозмірних об’єктів відбувається вдосконалення цього підходу.

Найпоширенішими представниками одноступеневих архітектур є YOLO (You Only Look Once) [13] та подібна до неї за принципом організації роботи SSD (Single Shot Detector) [14], а також їх численні модифікації. Такі модифікації представлені, зокрема**,** архітектурами Feature Pyramid Networks (FPN), [15] які є різновидом мережі типу SSD, що завдяки особливостям виявлення ознак краще ніж SSD розпізнають дрібні об'єкти, та RetinaNet [16].

Таким чином, основною особливістю одноступеневих підходів розпізнавання є те, що, на відміну від архітектур типу R-CNN, в яких виділення області зображення, яка потенційно може містити вартий уваги об’єкт, та її класифікація здійснюються різними нейронними мережами і, відповідно, згорткова мережа тут кілька разів застосовується до різних регіонів, які аналізуються. У випадку YOLO, SSD та їх модифікацій за знаходження об’єктів, визначення їх розміру та класу відповідає одна і та ж нейронна мережа. Плюсом також є те, що мережа переглядає все зображення відразу, враховує його вміст і при детектуванні, і при розпізнаванні об'єкта, що дозволяє суттєво пришвидшити швидкість роботи всієї системи розпізнавання. До недоліків підходу можна віднести зниження точності локалізації виявлених об'єктів в порівнянні з двоступеневими детекторами, особливо малорозмірних. Для усунення цього недоліку в роботах [17-19] було запропоновано удосконалення методу, яке значно покращило можливості розпізнавання саме таких об’єктів.

На підставі проведеного аналізу для виконання поставленої задачі було обрано архітектуру YOLO, яка дозволяє проводити розпізнавання візуальних об’єктів в реальному режимі часу, зокрема – її модифікацію YOLOv3 [17]. Використання цієї архітектури дозволяє проводити розпізнавання з досить високим рівнем вірогідності, у т. ч. і в реальному часі зі швидкодією понад 30 кадрів на секунду. Точність розпізнавання і швидкість процесу змінюються в залежності від розміру зображення, яке подається на вхід мережі.

Особливості архітектури та організація розпізнавання в YOLOv3

Основна ідея, покладена в основу роботи архітектури YOLO – на основі відповідних ознак розбити досліджуване зображення на $S \times S$-сітку прямокутних комірок, виходячи з припущення, що кожна така комірка є центром якогось об’єкту. Для кожної з цих комірок обчислюються п’ять параметрів: висота h та ширина w для трьох різних обмежуючих прямокутників, *Confidence* (вказує на величину ймовірності того, що дана комірка є центром якогось об’єкта) та *Class probability* (вказує на ймовірність того, що дана комірка відноситься до певного класу). Використовуючи значення цих параметрів, система проводить виділення та класифікацію об’єктів, присутніх на зображенні, яке аналізується.

YOLOv3 – вдосконалена версія архітектури YOLO, основна особливість якої полягає в тому, що для розпізнавання тут використовуються три шкали, кожна з яких розрахована на виявлення об'єктів різного розміру. Загалом, для розпізнавання у YOLOv3 використовується 106 шарів.

Виділення ознак та побудова відповідних карт ознак для зображення у YOLOv3 здійснюється з використанням загорткової мережі Darknet-53 [20], на вхід якої подаються зображення певного розміру (зазвичай 416×416 пікселів; однак, можуть подаватися і зображення інших розмірів). Відповідні карти ознак задаються шляхом зменшення розмірів вхідного зображення у 32, 16 та 8 разів і реєструються на виході 82, 94 та 106 шарів мережі, відповідно (рис. 1) [21].

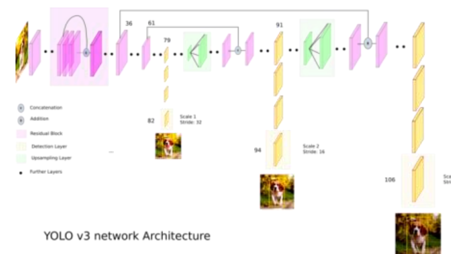


Рис. 1. Архітектура YOLOv3 [21]

Особливістю мережі Darknet є відмова від використання в ній агрегувальних шарів з операцією *max-pooling*; замість цього застосовується операція згортки з використанням фільтра, що містить лише одиниці, з кроком згортки, рівним 2. В результаті такої згортки також отримується зображення, зменшене в 2 рази, як і після операції *max-pooling*; але у Darknet отримується більш повна інформація на виході за рахунок кращої і повнішої передачі інформації між початковими та більш глибокими шарами нейронної мережі. Зазначається, що завдяки цьому і навчання мережі відбувається ефективніше [22]. Крім того, такий підхід дозволяє ефективніше вирішувати також задачі знаходження об'єктів на різних рівнях зображення (*image-level, region-level, pixel-level*) [23].

Іншою особливістю Darknet-53 є використання залишкових шарів (*Residual blocks*), які за рахунок специфічних зв'язків (*skip connections*) допомагають перенести дані з одного шару на інший, пропускаючи кілька проміжних [24]. Використання таких зв'язків дозволяє враховувати у глибоких мережах слабкі сигнали нейронів, які без такої передачі швидко "обнулюються", через що втрачаються різні дрібні деталі аналізу і збільшується величина його похибки.

Отримана на виході мережі Darknet-53 карта ознак надалі використовується для підрахунку значень параметрів *BBh* (*bounding box height*), *BBw* (*bounding box wight*), *Confidence* та *Class Probability* для кожного її елементу та наступного використання цих параметрів для розпізнавання об'єктів.

Подальша обробка зображення в YOLOv3 здійснюється з використанням підходу *Feature Pyramid Network* [25], у рамках якого розпізнавання об'єктів відбувається не на одній результуючій карті ознак, а на декількох, з різними розмірами. Використання у розпізнаванні карт ознак трьох розмірів – 13×13 (для великих об'єктів), 26×26 (об'єктів середнього розміру) та 52×52(дрібних об'єктів) дозволяє здійснювати надійне розпізнавання на зображенні об'єктів різних розмірів з уникненням втрати інформації про невеликі об'єкти.

Після проходження однією з карт ознак через розпізнавальний шар, до кожного її елемента застосовується "ядро виявлення" (*detection kernal*), на виході якого з'являється результуючий тензор, висота і ширина якого дорівнюють розміру карти ознак, а глибина рівна глибині "ядра виявлення". Глибина цього ядра обчислюється як добуток $b\times(5+c)$, де b – кількість обмежуючих прямокутників для кожного елемента (у нашому випадку їх 3), c – кількість класів, які ми розпізнаємо, 5 – кількість параметрів, які використовуються при розпізнаванні. До останніх відносяться: t_x та t_y – задають координати центра об'єкта; t_w та t_h – характеризують його розмір (ширину та висоту відповідно); p_0 – ймовірність знаходження об'єкта в даному елементі. Таким чином, для кожного елемента карти ознак ми отримуємо інформацію про три обмежуючі прямокутники (по одному для трьох різних масштабів розпізнавання) для кожного елемента вхідної матриці, ймовірність знаходження центру об'єкта у даній комірці та ймовірність його належності до певного класу об'єктів у вигляді результуючого тензора.

Оскільки для кожного елемента усіх трьох карт ознак мережа знаходить по 3 обмежуючі прямокутники, то загалом при розпізнаванні в сумі на виході ми маємо 10647 обмежуючих прямокутників; не всі з них вказують на об'єкт, що розпізнається. Щоб відсіяти усі хибні та залишити лише потенційно правильні рішення, використовується метод *Non-maximum suppression* (NMS) [26], суть якого полягає у встановленні певного порогового значення (*threshold*) ймовірності знаходження шуканого об'єкта у виділеному елементі та відкидання усіх прямокутників, для яких розраховане значення p_0 є меншими. Таким чином відсіюються усі точно хибні варіанти, які мають малі значення ймовірності знаходження в них шуканого об'єкта.

Далі вибираються усі прямокутники, що пересікаються між собою. З них вибирається один із найбільшим значенням p_0 і вираховується значення подібності (*intersection over union, IoU*), яке дозволяє оцінити, наскільки подібні між собою два варіанти. Значення параметра *IoU* для двох обмежуючих прямокутників визначається

як результат ділення площі їх перетину на площу їх об'єднання:

$$IoU=(BB1\wedge BB2)/(BB1\vee BB2),$$

де *BB1* – перший, а *BB2* – другий обмежуючий прямокутник. Отриманий результат порівнюють із попередньо встановленим пороговим значенням; якщо він вищий від цього значення, то даний варіант відкидається. Проведення такої операції для кожного виявленого об'єкта дозволяє позбутися хибних визначень і правильно визначати виявлені об'єкти.

Реалізація та навчання мережі

При реалізації програми було використано програмну імплементацію архітектури YOLOv3, виконану з використанням фреймворків TensorFlow та Keras авторами [27] і яка є у вільному доступі.

Для досягнення поставленої у роботі мети (розпізнавання об'єктів п'яти класів) була здійснена модифікація цієї архітектури шляхом заміни її вихідного шару, проведене навчання модифікованої таким чином мережі з відповідним аналізом особливостей самого процесу навчання та отриманих при його здійсненні результатів, вибір на їх основі оптимальних значень ваг та тестування роботи навченої мережі.

При навчанні модифікованої мережі як початкові були використані наявні у вільному доступі ваги навченої мережі, отримані для Darknet-53 [20]. Навчання мережі проводилося з використанням набору *Agrilfruit Dataset* [28] із застосуванням трьох різних варіантів значень параметрів градієнтного спуску. Параметри навченої мережі, які показали найкращі результати в процесі навчання, були застосовані при тестуванні її роботи.

Як уже згадувалося, для проведення навчання нейронної мережі був обраний *Agrilfruit Dataset*, який містить зображення п'яти видів фруктів – яблук, груш, лимонів, хурми та цукрових яблук. Загалом, у датасеті міститься 1500 зображень, по 300 зображень для кожного класу; з них для кожного класу 240 зображень використовувалися для проведення навчання мережі, а решта 60 – для її тестування.

Оскільки основним завданням навчання мережі з архітектурою YOLO є навчити її визначати, чи являється певна виділена комірка карти ознак центром шуканого об'єкту і, якщо це так, то який можливий розмір цього об'єкту та до якого класу він відноситься, для його проведення у YOLO необхідний набір даних певного визначеного формату. У якому, поряд із зображенням, для кожного об'єкта датасету має бути надана відповідна анотація з інформацією про координати лівого верхнього та правого нижнього кутів обмежуючого прямокутника, а також вказаний клас, до якого відноситься цей об'єкт.

У вигляді, в якому він надається користувачам, *Agrilfruit Dataset* містить безпосередньо самі зображення та текстові анотації до них, які представленні двома форматами – XML (містить абсолютні координати об'єктів та інформацію про їх класи) та TXT (містить нормалізовані відносні координати об'єктів та індекси класів об'єктів). Однак, жоден з цих форматів опису не підходить для навчання нейронної мережі на базі архітектури YOLO, яке вимагає специфічного опису об'єкта у *txt*-форматі. Тому перед її навчанням необхідно було підготувати набір даних у потрібному форматі. Для цього був використаний скрипт для конвертації XML-формату анотації у той, який вимагає архітектура YOLO. На виході даний скрипт формує наступні 3 файли: *Names.txt* – містить назви класів; *Train.txt* – містить анотації для зображень, які будуть використані для навчання; *Test.txt* – містить анотації для зображень, які будуть використані для оцінки точності результату.

Під час навчання мережі була використана методика *warm-up steps* [29], суть якої полягає у тому, що, з метою уникнення нестабільності процесу навчання, для початкової епохи навчання значення коефіцієнта швидкості навчання є дуже малою і поступово збільшується під час декількох наступних епох навчання (які називають *warm-up*-епохами*)* до заданого початкового значення.

Навчання модифікованої мережі проводилося з використанням ПК з оперативною пам'яттю 8 Gb DDR3 та процесором Intel Core i5-2500K 3.30GHz×4 зі встановленою ОС Ubuntu 18.04.05 LTS.

Для навчання мережі використовувалися такі гіперпараметри:

– кількість епох – визначає кількість циклів навчання;
– *Batch size* – вказує розмір батчу, який використовується на одному кроці навчання;
– кількість *warm-up* епох – визначає кількість циклів, на протязі яких коефіцієнт навчання буде зростати від 0 до необхідного початкового значення;
– початковий (*learning rate*) та кінцевий коефіцієнти навчання;
– *Score threshold* – порогове значення для результатів передбачення наявності центра об'єкта у певній комірці зображення, значення нижче якого не беруться до уваги;
– *IoU threshold* – порогове значення для міри подібності об'єктів.

Було проведено навчання отриманої мережі з використанням трьох варіантів застосування градієнтного спуску (SGD) – стохастичного та мінібатчевих зі значенням кількості предметів у батчі 4 та 8, відповідно. Результати проведеного навчання такі:

1. При застосуванні SGD (з такими значеннями гіперпараметрів: кількість епох навчання – 20; кількість w*arm-up* епох – 4; Batch size – 1; значення початкового та кінцевого коефіцієнтів навчання – 1×10^{-4} та 1×10^{-6}; *Score threshold* – 0.3; *IoU threshold* – 0.5) – тривалість процесу навчання склала 22 год. 33 хв; отримане середнє значення точності визначення *mAP* (*mean average precision*) – 66.40%.
2. У другому випадку – (кількість епох навчання – 10; кількість *warm-up* епох – 2; *Batch size* – 4) процес навчання зайняв 9 год. 32 хв., а значення *mAP*=87.01%.
3. Для третього (більшість обраних параметрів була ідентичною до попереднього, за виключенням *Batch size* – 8) – процес навчання зайняв 9 год. 12 хв.; значення *mAP* рівне 82.98%.

Графіки зміни величини похибки, отриманої в процесі навчання мережі у трьох описаних випадках, показані на рис. 2.

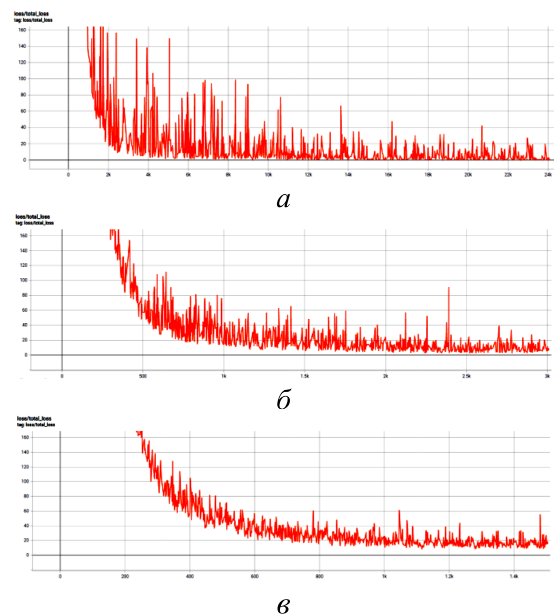


Рис. 2. Величина похибки навчання, отриманої при навчанні з різними значеннями гіперпараметрів навчання з використанням SGD з розмірами *Batch size*: *а* – 1; *б* –4; *в* – 8

Після кожної епохи навчання здійснювалася оцінка якості навчання мережі на тестовій частині датасету. Значення відповідних похибок, отримані в процесі тестування мережі для вказаних випадків, показані на рис. 3.

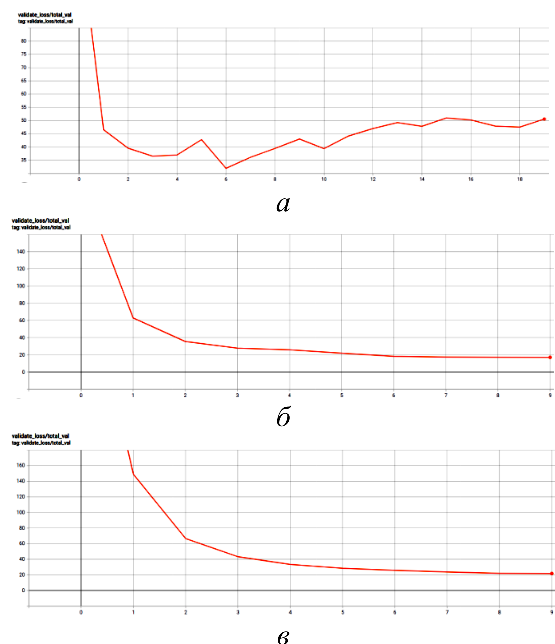


Рис. 3. Величина похибки, отриманої при тестуванні з різними значеннями гіперпараметрів навчання з використанням SGD з розмірами *Batch size*: *а* – 1; *б* –4; *в* – 8

Як видно з рис 3.*а*, для стохастичного SGD (*Batch size*=1), найнижче значення похибки для тестових даних було отримане після виконання 6 епохи навчання, після чого значення похибки почало зростати. Це може свідчити про те, що мережа починає “перенавчатися”.

У випадку з *Batch size*, рівним 4 (рис. 3,*б*), після шостої епохи наступні епохи навчання вносять досить незначні похибки у точність визначення. Подібний характер зміни похибки спостерігався і у випадку з *Batch size*, рівним 8 (рис. 3,*в*), з тією відмінністю, що навчання тут на початкових епохах відбувається більш плавно, а після шостої епохи зміна похибки є навіть дещо більшою, ніж у другому випадку.

З умови вибору мінімального значення середньої похибки розпізнавання були вибрані оптимальні значення ваг мережі – у нашому випадку вони відповідають вагам, отриманим при навчанні зі значеннями *Batch size*, рівними 4. Які й були використані у подальшій роботі з програмою при практичному розпізнаванні об’єктів на пред’явлених зображеннях.

Робота програми

Реалізована програма має графічний інтерфейс, побудований за допомогою бібліотеки *TKinter*. Вигляд інтерфейсу показаний на рис. 4.

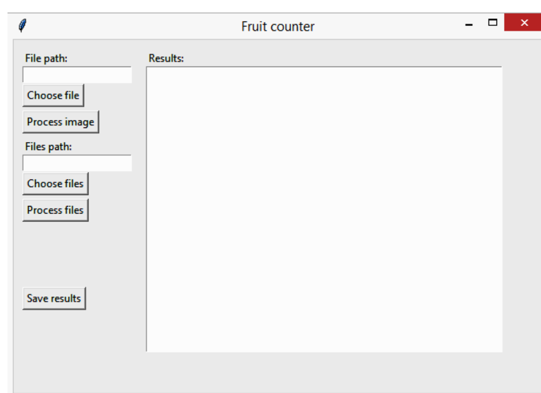


Рис. 4. Графічний інтерфейс програми

Програма має два режими роботи – з безпосереднім виведенням результату та фоновий.

У першому режимі роботи необхідно вибрати вхідне зображення, яке буде аналізуватися. Зробити це можна, натиснувши на кнопку “Choose file” та вибравши зображення у відповідному вікні, що відкриється (вихідне зображення може бути будь-якого розміру). Після вибору зображення, адреса вибраного файлу буде відображена у полі “File path:” (рис. 4). Адресу для зчитування потрібного файлу можна у це поле ввести й самостійно.

Для проведення розпізнавання, необхідно натиснути на кнопку “Process image”. У даному режимі роботи програми після закінчення розпізнавання буде отримане результуюче зображення, у якому біля кожного виявленого об’єкта, обмеженого квадратом, виведена його назва та достовірність віднесення до даного класу. Програма також виведе текстову інформацію про результат та аналізоване зображення, на якому будуть помічені розпізнані об’єкти.

Приклад результату роботи програми у цьому режимі зображено на рис. 5.

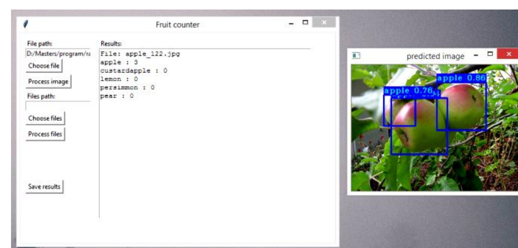


Рис. 5. Результат виконання програми у графічному режимі

У другому режимі роботи програма може опрацьовувати декілька файлів у фоновому режимі, без виведення кінцевого результату на екран. Для цього необхідно натиснути на кнопку “Choose files”, після чого вибрати файли з потрібними зображеннями у вікні, що відкриється (вигляд вікна програми при виборі потрібних зображень для фонового аналізу показаний на рис. 6). Адреси вибраних файлів будуть виведені у полі під надписом “Files path”. Також адреси потрібних для аналізу файлів можна ввести у вище згадане поле, використовуючи пробіл як розділювач для адрес.

Для запуску процесу аналізу необхідно натиснути на кнопку “Process files”, після чого вибрані зображення будуть опрацьовані, а результати будуть виведені у текстовому полі “Results:”.

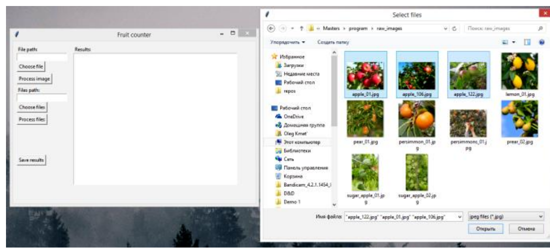


Рис. 6. Вибір декількох файлів для фонового аналізу

Результати фонового розпізнавання двох зображень з яблуками та грушами показані на рис. 7. Як бачимо, окрім результатів для кожного зображення, у текстовому полі “Results:” дається також загальний підсумок кількості знайдених на обох зображеннях фруктів.

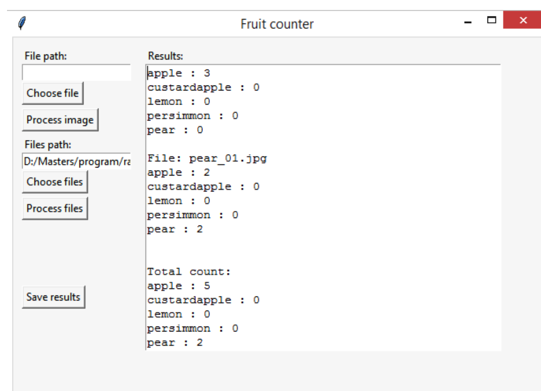
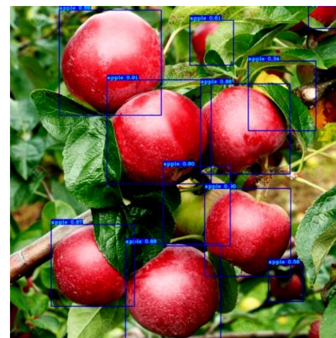


Рис. 7. Результат роботи програми у фоновому режимі

Для збереження результату необхідно натиснути на кнопку “Save results” та вибрати директорію і назву файлу у вікні, що відкриється. Після чого програма збереже увесь текстовий вивід у файл, а також усі опрацьовані зображення у папку “Processed images”.

Проведені тестування роботи програми показали, що вона з досить хорошою достовірністю визначає відповідні об’єкти в широкому інтервалі їх розмірів на зображенні навіть у випадку, коли їх зображення не повні або частково прикриті. Як можна бачити на рис. 8, у випадку розпізнавання яблук, програма добре справляється з поставленою задачею, розпізнаючи з високою достовірністю (понад 80%) добре помітні яблука різних розмірів та зі значною долею достовірності (понад 50%) такі, велика частина яких або закрита листям, або має суттєві розмитості чи незначні розміри.

Аналогічні результати були отримані при розпізнаванні зображень, які містять всі п’ять видів фруктів, для яких було проведено навчання мережі. Крім того, програма працює дуже швидко, проводячи аналіз пред’явленого зображення за долі секунди, фактично в режимі реального часу. Що дозволяє використовувати даний застосунок для експресного підрахунку кількості плодів на зображеннях дерева. Таким чином, програма дає можливість використовувати її для проведення приблизного експрес-аналізу врожаю.



```
File:
apple_02.jpg
apple : 11
custardapple : 0
lemon : 0
persimmon : 0
pear : 0
```

Рис. 8. Результати розпізнавання зображення з яблуками

Висновки

В статті представлені реалізація та особливості застосування програми на основі нейромережі, навченої розпізнавати фрукти п’яти класів. При її створенні за основу було використано нейромережеву структуру YOLOv3 в її імплементації в *Tensorflow* 2.1 з відповідною модифікацією вихідного шару. Донавчання створеної таким чином мережі здійснено на персональному комп’ютері з загальнодоступними параметрами, застосовуючи технологію *Transfer learning* з використанням значень ваг попередньо навченої мережі, наявних у вільному доступі, та датасету *Agrilfruit Dataset*.

Проведені дослідження особливостей процесу навчання за умови використання градієнтного спуску з різними параметрами використання датасету (стохастичного та мінібатчевих з різними значеннями кількості об’єктів у батчі) дозволили встановити вплив умов навчання на його проходження та обрати оптимальний варіант ваг навченої мережі для подальшого її використання.

Здійснене тестування роботи програми показало, що створений додаток з високою достовірністю розрізняє об’єкти різних розмірів на зображенні та підраховує їх кількість.

Реалізовану програму можна використовувати у будь-якому програмному середовищі, яке містить інтерпретатор мови Python. Оскільки застосунок дає можливість ідентифікувати та підраховувати кількість окремих фруктів на аналізованому зображенні, він може бути використаний для візуальної оцінки врожаю фруктових дерев.

Дана програма може легко бути модифікована для розпізнавання об’єктів інших класів. Для цього потрібно підібрати відповідний начальний датасет та провести перенавчання моделі.

References

1. Licheng Jiao et al. A Survey of Deep Learning-based Object Detection. // arXiv:1907.09408v2 [cs.CV] 10 Oct 2019.
2. Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. // arXiv:1905.05055v2 [cs.CV] 16 May 2019.
3. P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art // IEEE Transactions on Pattern Analysis and Machine Intelligence – 2012, vol. 34 – pp. 743–761.
4. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite // IEEE Conference on Computer Vision and Pattern Recognition, 2012 – pp. 3354–3361.
5. O. Russakovsky et al. Imagenet large scale visual recognition challenge // International Journal of Computer Vision – 2015, vol. 115. – pp. 211–252.
6. M. Everingham et al. The pascal visual object classes (voc) challenge // International Journal of Computer Vision – 2010, vol. 88. – pp. 303–338.
7. Lin T.-Y. et al. Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham // arXiv:1405.0312v3 [cs.CV] 21 Feb 2015.
8. Kuznetsova, H. Rom, N. Alldrin et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale // arXiv:1811.00982, 2018.
9. Pogruzheniye v svortochnyye neyronnyye seti: peredacha obucheniya (transfer learning) habr.com. 2019. Available: https://habr.com/ru/post/467967/.
10. R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Tech report (v5) // arXiv:1311.2524v5 [cs.CV] 22 Oct 2014.
11. Girshick R. Fast R-CNN. // arXiv:1504.08083v2 [cs.CV] 27 Sep 2015.
12. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // arXiv:1506.01497v3 [cs.CV] 6 Jan 2016.
13. J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection // arXiv:1506.02640v1 [cs.CV] 8 Jun 2015.
14. W. Liu A et al. SSD: Single shot multibox detector. In: Computer Vision – ECCV 2016 (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), Springer International Publishing, 2016. – pp. 21–37. // arXiv:1512.02325v5 [cs.CV] 29 Dec 2016.
15. Tsung-Yi Lin et al. Feature Pyramid Networks for Object Detection // arXiv:1612.03144v2 [cs.CV] 19 Apr 2017.
16. Y. Li and F. Ren. Light-Weight RetinaNet for Object Detection // arXiv:1905.10011v1 [cs.CV] 24 May 2019.
17. Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. // arXiv:1804.02767v1 [cs.CV] 8 Apr 2018.
18. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection // arXiv:2004.10934v1 [cs.CV] 23 Apr 2020.
19. Delong Qi, Weijun Tan, Qi Yao, Jingfeng Liu. YOLO5Face: Why Reinventing a Face Detector. // arXiv:2105.12931v1 [cs.CV] 27 May 2021.
20. J. Redmon. Darknet: Open source neural networks in C. Available: http://pjreddie.com/darknet/.
21. Kathuria A. What's new in YOLOv3? Towar. Data Sci., 2018. Available: https://towardsdatascience.com/yolov3-object-detection-53fb7d3bfe6b.
22. J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller. Striving for simplicity. The all convolutional net. // arXiv.1412.6806v3 [cs.LG] 13 Apr 2015.
23. Shuyang Sun et al. FishNet. A Versatile Backbone for Image, Region, and Pixel Level Prediction // arXiv:1901.03495v1 [cs.CV] 11 Jan 2019.
24. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition / arXiv:1512.03385v1 [cs.CV] 10 Dec 2015.
25. T.-Y. Lin et al. Feature Pyramid Networks for Object Detection // arXiv:1612.03144v2 [cs.CV] 19 Apr 2017.

26. Jan Hosang Rodrigo Benenson Bernt Schiele. Learning non-maximum suppression // arXiv:1705.02950v2 [cs.CV] 9 May 2017.
27. TensorFlow-2.x-YOLOv3 and YOLOv4 tutorials: 2020. Available: https://github.com/pythonlessons/TensorFlow-2.x-YOLOv3.
28. Pawara P. Agrilfruit Dataset – for object detection and counting task Pornntiwa Pawara // ai.rug.nl. – 2020. Available: https://www.ai.rug.nl/~p.pawara/dataset.php.
29. Junyuan Xie Tong et al. Bag of Tricks for Image Classification with Convolutional Neural Networks. // arXiv:1812.01187v1 [cs.CV] 4 Dec 2018.

Література

1. Licheng Jiao et al. A Survey of Deep Learning-based Object Detection. // arXiv:1907.09408v2 [cs.CV] 10 Oct 2019.
2. Z. Zou, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey. // arXiv:1905.05055v2 [cs.CV] 16 May 2019.
3. P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art // IEEE Transactions on Pattern Analysis and Machine Intelligence – 2012, vol. 34 – pp. 743–761.
4. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite // IEEE Conference on Computer Vision and Pattern Recognition, 2012 – pp. 3354–3361.
5. O. Russakovsky et al. Imagenet large scale visual recognition challenge // International Journal of Computer Vision – 2015, vol. 115. – pp. 211–252.
6. M. Everingham et al. The pascal visual object classes (voc) challenge // International Journal of Computer Vision – 2010, vol. 88. – pp. 303–338.
7. Lin T.-Y. et al. Microsoft COCO: Common Objects in Context. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8693. Springer, Cham // arXiv:1405.0312v3 [cs.CV] 21 Feb 2015.
8. Kuznetsova, H. Rom, N. Alldrin et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale // arXiv:1811.00982, 2018.
9. Погружение в свёрточные нейронные сети: передача обучения (transfer learning) habr.com. 2019. Available: https://habr.com/ru/post/467967/.
10. R. Girshick, J. Donahue, T. Darrell, J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. Tech report (v5) // arXiv:1311.2524v5 [cs.CV] 22 Oct 2014.
11. Girshick R. Fast R-CNN. // arXiv:1504.08083v2 [cs.CV] 27 Sep 2015.
12. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks // arXiv:1506.01497v3 [cs.CV] 6 Jan 2016.
13. J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection // arXiv:1506.02640v1 [cs.CV] 8 Jun 2015.
14. W. Liu A et al. SSD: Single shot multibox detector. In: Computer Vision – ECCV 2016 (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), Springer International Publishing, 2016. – pp. 21–37. // arXiv:1512.02325v5 [cs.CV] 29 Dec 2016.
15. Tsung-Yi Lin et al. Feature Pyramid Networks for Object Detection // arXiv:1612.03144v2 [cs.CV] 19 Apr 2017.
16. Y. Li and F. Ren. Light-Weight RetinaNet for Object Detection // arXiv:1905.10011v1 [cs.CV] 24 May 2019.
17. Joseph Redmon, Ali Farhadi. YOLOv3: An Incremental Improvement. // arXiv:1804.02767v1 [cs.CV] 8 Apr 2018.
18. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection // arXiv:2004.10934v1 [cs.CV] 23 Apr 2020.
19. Delong Qi, Weijun Tan, Qi Yao, Jingfeng Liu. YOLO5Face: Why Reinventing a Face Detector. // arXiv:2105.12931v1 [cs.CV] 27 May 2021.
20. J. Redmon. Darknet: Open source neural networks in C. Available: http://pjreddie.com/darknet/.
21. Kathuria A. What's new in YOLOv3? Towar. Data Sci., 2018. Available: https://towardsdatascience.com/yolov3-object-detection-53fb7d3bfe6b.
22. J. T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller. Striving for simplicity. The all convolutional net. // arXiv.1412.6806v3 [cs.LG] 13 Apr 2015.
23. Shuyang Sun et al. FishNet. A Versatile Backbone for Image, Region, and Pixel Level Prediction // arXiv:1901.03495v1 [cs.CV] 11 Jan 2019.
24. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition / arXiv:1512.03385v1 [cs.CV] 10 Dec 2015.
25. T.-Y. Lin et al. Feature Pyramid Networks for Object Detection // arXiv:1612.03144v2 [cs.CV] 19 Apr 2017.
26. Jan Hosang Rodrigo Benenson Bernt Schiele. Learning non-maximum suppression // arXiv:1705.02950v2 [cs.CV] 9 May 2017.
27. TensorFlow-2.x-YOLOv3 and YOLOv4 tutorials: 2020. Available: https://github.com/pythonlessons/TensorFlow-

2.x-YOLOv3.
28. Pawara P. Agrilfruit Dataset – for object detection and counting task Pornntiwa Pawara // ai.rug.nl. – 2020. Available: https://www.ai.rug.nl/~p.pawara/dataset.php.
29. Junyuan Xie Tong et al. Bag of Tricks for Image Classification with Convolutional Neural Networks. // arXiv:1812.01187v1 [cs.CV] 4 Dec 2018.

Received 17.10.2021
Accepted 27.11.2021